Implementing metaheuristic optimization algorithms with JECoLi

Pedro Evangelista[1,2] Paulo Maia[1,2]
Miguel Rocha[1]
[1]CCTC - Computer Science and Technology Center
[2]CEB - Centre of Biological Engineering / IBB
Universidade do Minho
Campus de Gualtar, Braga, Portugal
{paulo.maia, ptiago}@deb.uminho.pt, mrocha@di.uminho.pt

Abstract

This work proposes JECoLi - a novel Java-based library for the implementation of metaheuristic optimization algorithms with a focus on Genetic and Evolutionary Computation based methods. The library was developed based on the principles of flexibility, usability, adaptability, modularity, extensibility, transparency, scalability, robustness and computational efficiency. The project is open-source, so JECoLi is made available under the GPL license, together with extensive documentation and examples, all included in a community Wiki-based web site (http://darwin.di.uminho.pt/jecoli). JECoLi has been/is being used in several research projects that helped to shape its evolution, ranging application fields from Bioinformatics, to Data Mining and Computer Network optimization.

1. Introduction

Over three decades ago, in both sides of the Atlantic, a number of researchers were developing new approaches to tackle hard optimization problems. Those had in common their inspiration, since they were based on analogies with the way natural creatures solve their own optimization problems: natural selection. This trend led to the development, in the 1960s/70s, of methods such as Genetic Algorithms (GA) [5], Evolution Strategies (ES) [18], Evolutionary Programming (EP) [4] or Genetic Programming (GP) [8].

Over the last decades, these methods have been used to solve numerous problems over a wide range of scientific and technological fields, with an overall remarkable success. Also, a large number of variants have been proposed (e.g. Differential Evolution [21] or Memetic Algorithms [12]), new nature-based approaches were developed (e.g. Particle Swarms [6]) and new features were introduced (e.g. new solution encoding schemes, selection or reproduction operators).

All this activity lead to an impressive growth of the whole field of Genetic and Evolutionary Computation (GEC), easily measured by the number of scientific publications, books, international conferences and journals, subjects/ courses taught at universities, commercial applications, etc. Also, a number of open-source software platforms have been provided with implementations of sub-sets of these methods (see Table 1 for some examples). These aim to take into account the common features underlying these approaches, to create software components that can be used by developers to implement their applications and ultimately solve their optimization problems.

However, none of these platforms was able to achieve a significant success in terms of the number of users, being

Table 1. Examples of some general purpose software platforms for GEC.

Name	URL
Opt4J	http://opt4j.sourceforge.net
JGAP	http://jgap.sourceforge.net
ECJ	http://cs.gmu.edu/ eclab/projects/ecj/
JAGA	http://www.jaga.org
Eva2	http://www.ra.cs.uni-tuebingen.de/software/EvA2
JCLEC	http://jclec.sourceforge.net/
Watchmaker	https://watchmaker.dev.java.net/
GAA	http://www.aridolan.com/ga/gaa/gaa.html
PISA	http://www.tik.ee.ethz.ch/sop/pisa/
EO	http://eodev.sourceforge.net/
OpenBeagle	http://beagle.gel.ulaval.ca/
JECL	http://sourceforge.net/projects/jecl

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.161

normally restricted to a few research groups and specializing in some domains of application. This situation can be explained by the lack of flexibility or adaptability of the tools, given the broad range of requirements imposed by the users. Also, in some cases, the lack of appropriate documentation that can help to ease the learning curve is also noticed. Therefore, we believe there is still room for the proposal of new software platforms in this field.

This paper introduces the Java Evolutionary Computation Library (JECoLi), a novel framework that allows the development of metaheuristic optimization algorithms, using the Java programming language. JECoLi is focused on implementing approaches from the GEC field, while supporting other popular alternative methods.

It aims to be an adaptable, flexible, extensible and modular software platform, supporting two main types of tasks: (i) to develop components of other systems and applications using the provided optimization algorithms; (ii) to allow the rapid benchmarking of distinct approaches in specific optimization tasks.

The main features of the library, together with some issues regarding its architecture are given in the next section. Then, some examples and research projects that use JECoLi are briefly explained. The paper finishes with the conclusions and further work.

2. Architecture and main features

2.1. Main features

Available algorithms JECoLi implements a large set of metaheuristic algorithms, namely:

- General purpose Evolutionary Algorithms, including Genetic Algorithms and Evolutionary Programming;
- Differential Evolution (variants DE/rand and DE/best variants);
- Genetic Programming and Linear GP;
- Simulated Annealing (SA) [7];
- Cellular Automata GAs [1];
- Multi-objective optimization Evolutionary Algorithms (NSGA II [2] and SPEA2 [23]);

For all these algorithms, a large number of variants can be executed, using the configuration classes for each case. These enable to setup the encoding scheme, reproduction and selection operators, the termination criteria and other specific parameters. To support beginner users, default parameters are provided in most cases.

Encoding schemes Solutions can be encoded using different types of representations: binary, integer, real, permutations, sets or trees are currently available. Numerous reproduction operators are provided for each of these representations. In the current release, a total of 29 reproduction operators (crossover and mutation) are available. Also, 8 distinct selection operators were implemented for EAs. A number of pre-processing schemes applicable to the selection operators are also made available (e.g. scaling, ranking).

Hybridization Adequate support is given to the combination of general purpose algorithms with problem-specific methods, therefore allowing the easy development of hybrid approaches. The main examples are:

- local optimization enriched EAs (also termed Memetic Algorithms), where local optimization operators are defined as mutation operators. The user can also opt to re-define the methods that run the algorithms and incorporate it as a mandatory step for all individuals.
- hybrid crossover operators that use information from the problem instance in the process of combining information from the parents;
- initial population enrichment with problem-specific heuristics.

Flexibility, adaptability, modularity and extensibility The platform is highly flexible, allowing the available components to be easily arranged and configured in diverse ways. A loose coupling is provided between optimization algorithms and problems that allows the easy integration of the library with other software. New problems are added by the definition of a single class (the evaluation function).

JECoLi provides a component-based platform, with a set of reusable modules that can be used by developers to build applications and integrating new components. In fact, a number of programming interfaces are provided that enable the extension of the platform with novel algorithms, representations, reproduction or selection operators, termination criteria, etc. These are easily integrated into the existing platform and automatically profit from the available features.

Compatibility and software development process The code is 100% Java, using the most recent features (e.g. generics) to make the development simple while enforcing good development practices. In the development process, the issues of computational efficiency, robustness and scalability (e.g error handling, software testing) were taken into account.

Availability, documentation and web site The framework is open source released under the GPL license, allowing its easy integration into other projects. Extensive documentation (several howto's, examples and a complete API reference), binaries and source code releases are available in the Wiki-based project's web site (http://darwin.di.uminho.pt/jecoli)

2.2. Architecture

In Figure 2.2 a simplified class diagram of the library is shown. For easier legibility this contains only the main entities of the software. The general software structure contains the following set of entities:

- Algorithm: represents the optimization method abstraction, i.e. all algorithms share a number of features that are summarized in the *IAlgorithm* interface.
- Termination Criteria: verifies if the termination conditions are satisfied. Several alternative criteria that follow the *ITerminationCriteria* interface are defined (e.g. number of generations, function evaluations, computational time, target fitness value) although the developer has the option to create new ones.
- Evaluation Function: represents the classes that handle the decoding of the genomes into solutions to the problem and the fitness assignment process. This component is dependent on the problem domain and makes the connection between the problem and the algorithm's domains. Each instance of an evaluation function implements the *IEvaluationFunction* interface.
- Algorithm Configuration: contains all the necessary information to execute the different algorithms and can be further divided in two separate components:
  - Information common to all algorithms: termination criteria, evaluation function and statistics configuration (specifies the metrics to be calculated and displayed on the screen and/or to be saved to disk).
  - Algorithm dependent information: the different algorithm parameters (e.g. selection operators or recombination parameters in EAs, or the annealing schedule in SA).
- Solutions and solution sets: each solution is composed by a genome encoding it in a specific representation and a set of fitness values (one for each objective), following the *ISolution* interface. Populations and archives are implemented by the *ISolutionSet* interface, that keeps lists of solutions with enhanced functionalities.
- Representations: The classes that implement solution representation follow the *IRepresentation* interface. Its hierarchy allows to work over logical representations that do not depend on the underlying data structures. For example a linear genome can be portrayed by a linked list or a vector, if there exists a class that implements the *ILinearRepresentation* interface.
- Solution factories: To build and copy solutions, the concept of solution factories was developed. This notion is based on the factory design pattern. In certain situations, these factories also serve as central repositories of constraints applied to a certain genome, e.g. in a real value representation each gene has a lower and upper limit. Auxiliary methods, like generating a specific gene, were also created to ease the programming burden. These factories are extensively used by the predefined set of reproduction operators available.

If the necessary parameters have been instantiated, the algorithm can be executed. While running, each algorithm has an internal state that represents the algorithm status and the set of results collected during previous iterations. The state includes the current solution set being manipulated by the algorithm, previously obtained solutions (based on user defined criteria) and a set of iteration dependent statistics. This strategy allows to save the algorithm state independently from the method implementation and to decouple the method implementation from statistics specific code.

3. Examples and research projects

3.1. Examples

A number of examples is made available in the project's web site to illustrate the main features of the library. These are listed below:

- the *counting ones* task is used as a toy problem to show potential developers how to address the resolution of a problem with several algorithms.
- the classical Traveling Salesman Problem illustrates the use of permutation representations, as well as the development of problem specific mutation and crossover operators (i.e. hybrid approaches). It is also used to show the support to perform several runs of a given algorithm and configuration.
- the numerical optimization examples include several benchmarking functions and show the use of real value representations. The algorithms used are the EA, the SA and the DE.

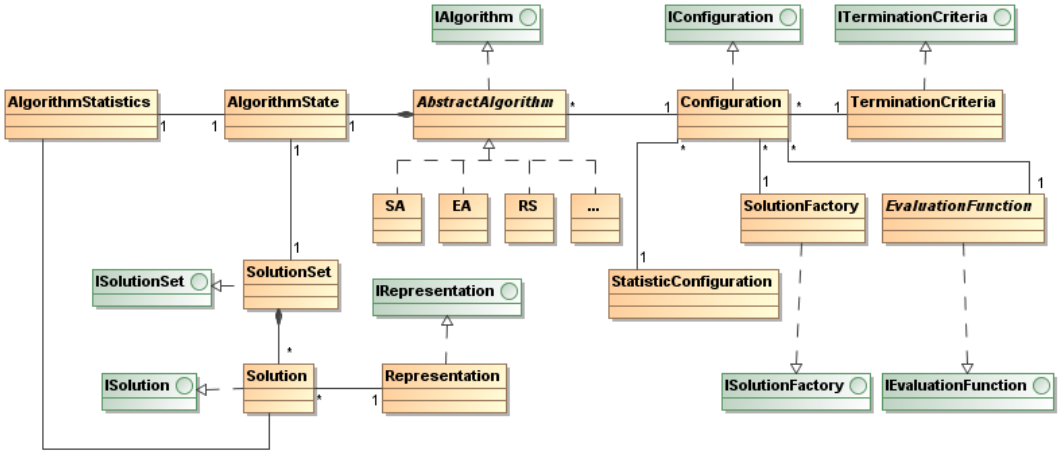

Figure 1. Class diagram of the general architecture of the framework

- the Knapsacking problem shows how to handle constraints in combinatorial optimization problems.
- the multiobjective optimization example shows the application of the NSGAII and SPEA2 algorithms.
- the symbolic regression task shows the use of GP and Linear GP.

3.2. Projects using JECoLi

In this section, we present an overview of the main research projects that have been using the JECoLi software over the last few years showing its maturity. In each case, we present a brief statement of the project's main goals, of the problem definition and also of the approach taken to solve them using JECoLi, revealing the main technical issues addressed in each case.

Metabolic engineering applications In this work, the main aim is to identify sets of gene deletions towards the maximization of a desired physiological objective function. The optimization problem consists in selecting, from a set of genes in a microbe's genome, a subset to be deleted in order to maximize a given objective function.

Our approach was to develop EAs and SA algorithms using a set-based representation, where only gene deletions are represented in the solution [14]. Since a variable size representation is used, this allows the automatic finding of the best number of gene deletions necessary for achieving a given productivity goal.

Each solution (mutant strain) is evaluated by resorting to the simulation of its phenotype using a steady-state approach. This allows to calculate the values of the fluxes for the organism's metabolic model. The adopted fitness function is the Biomass-Product Coupled Yield, a non linear function that aims to simultaneously maximize the biomass and the desired product fluxes. These algorithms have been implemented as part of the OptFlux software platform (http://www.optflux.org).

Recently, the development of methods for multi-objective optimization [10] and the simulation of mutants using dynamic models based on ordinary differential equations (ODEs) have also been approached [3].

Fermentation optimization In fed-batch fermentations there is an addition of nutrients along the process, allowing the achievement of higher product concentrations. During this process the systems states change considerably and this dynamic behavior motivates the development of optimization methods to find the optimal input feeding trajectories in order to improve the process performance. White box mathematical models based on ODEs that represent the mass balances of the relevant state variables are typically used to simulate the processes.

The numerical optimization task addressed in this work [11] aims at finding the best trajectory of some input variables (feeding), that yield the maximum performance index, defined in each specific case. A solution to the problem will consist of a set of real-valued vectors of equal length. Each vector encodes an input variable as a temporal sequence of values, defined as a piecewise linear function. Feeding values are provided only at equally spaced points, while the remaining values are linearly interpolated.

The evaluation process, for each solution, is achieved by running a numerical simulation of the defined model, given as input the feeding values. For any solution, the fitness value is then calculated after the simulation, taking the calculated values of the state variables, according to a performance index defined for each case.

To solve this problem, EAs with real value representations, several variants of DE and also PSO [6] were tested and evaluated. The DE/rand methods were the best candidates in most case studies. Also, in recent work [16] we have approached this optimization problem, but considering its online implementation (i.e. performed in real-time while the fermentation process is running).

Network traffic engineering In this project, the main aim is to improve the Quality of Service levels in TCP/IP based networks, by configuring the routing weights of link-state intra-domain routing protocols such as Open Shortest Path First (OSPF). A mathematical model is used to provide the simulation environment, allowing to define flexible cost functions that can take into account several measures of the network behaviour.

We aim to improve the process of OSPF weight setting implementing traffic engineering methods [19], assuming that the administrator has access to a matrix representing traffic demands between each pair of nodes in the network. The particular problem addressed in this work was to find the best configuration that simultaneously could minimize the overall network congestion and the end-to-end delays between the pairs of routers in the network.

Since the underlying problem is NP-hard, the approach was to develop EAs to solve it. In order to encode the solutions, integer representations were used, where each solution encodes a set of weights, one for each network link. Those are used to configure the OSPF routing protocol and therefore allow the calculation of the routing tables, using the well known Dijkstra algorithm.

The fitness assignment process goes through the simulation of the network behaviour taking into account the network model and the provided traffic demands matrix. Using this information, the loads in the network links and the overall end-to-end delays are calculated. In both cases, a penalty function is defined to assess the performance of the network in each objective.

Two approaches have been implemented to address this multi-criterion problem: (i) to use linear weighting functions, with a parameter that tunes the trade-off between both components of the cost function; and, (ii) to use multiobjective EAs, such as NSGAII or SPEA2.

More recently, this work has been extended to handle multi-class traffic [20] and also to provide traffic engineering approaches for networks with both unicast and multicast traffic [17].

Neural network evolution The search for the optimal Artificial Neural Network (ANN) to solve a particular problem is a challenging task: the ANN should learn the inputoutput mapping without overfitting the data and training algorithms may get trapped in local minima. This problem involves two main distinct tasks: selecting an appropriate topology (how many layers and neurons in each layer, which nodes are connected) and training the ANN, i.e. setting the adequate weights to the connections.

In our work [13] we proposed two hybrid GEC/ ANN algorithms: the first evolves neural topologies, while the latter performs simultaneous optimization of the architecture and weights. In both cases, a solution to the problem is directly encoded, i.e. the EA's individuals directly include the ANN and the reproduction operators work over the ANNs changing their topologies and connection weights.

In the first case, the evaluation of the solution involves training the encoded ANN and estimating the error over a set of validation examples. In the latter case, only the second operation is necessary.

Biomarker discovery DNA microarrays allow to measure the expression of all genes in a genome and is becoming quite important in biomedical research. In particular, the automatic classification of samples has been a promising approach in cancer diagnosis and a number of classifiers have been proposed.

A major problem with the application of these methods is the huge number of attributes (genes) in the datasets (typically thousands). Gene reduction is extremely important because it usually increases the accuracy of the classifiers and it provides sets of genes that are important for classification (e.g. biomarkers).

Our work [15] relies on the development of a wrapper feature selection approach based on EAs as the optimization engine, where solutions encode a set of features using a variable size set-based representation. The evaluation of each solution requires the use of a classifier (k-nearest neighbour, decision trees and support vector machines were used) and the estimation of its error (using methods of cross-validation or leave-one out); this process is achieved by using the Weka software system [22].

Artificial Life An Artificial Life environment (*getALife*) has been developed [9], whose major aim is to provide a framework to evaluate single and multi-agent systems and evolutionary approaches to the development of reinforcement learning algorithms.

The environment is based on a predator-prey scenario, with multiple species and where individuals are mainly characterized by their decision modules and genetic information. The encoding of the decision modules in the genome, as well as all the reproduction processes are handled by JECoLi software. A recent version of the getAlife GUI is provided in http://darwin.di.uminho.pt/alife.

4. Conclusions and further work

In this paper, a novel optimization library and the major architectural software decisions were presented. The main aim of this work was to provide a new and extensible framework, for the development of meta-heuristics and also to stimulate and complement the existing efforts in the software engineering community, to build better optimization packages, allowing these methods to reach a wider scientific audience.

Since this is still a young framework, most of the future work includes the improvement of the existing functionalities and support to the current users. Also, we aim to develop new capabilities including new algorithms (e.g. Evolution Strategies or Particle Swarms), new representations or selection/reproduction operators. Other future aims are to develop appropriate support for GUI development over the library and to implement wrappers that allow the integration of the framework with existing scientific computation environments (e.g. Matlab, R, etc.)

References

[1] E. Alba and B. Dorronsoro. *Cellular genetic algorithms*. Springer Verlag, 2008.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[3] P. Evangelista, I. Rocha, E. Ferreira, and M. Rocha. Evolutionary approaches for strain optimization using dynamic models under a metabolic engineering perspective. In C. Pizzuti and M. Ritchie, editors, *Proceedings of the EvoBio 2009, Lecture Notes Computer Science*, 2009.

[4] L. Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley and Sons, New York, 1999.

[5] J. Holland. *Adaptation in Natural and Artificial Systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.

[7] S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[8] J. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992.

[9] D. Machado and M. Rocha. getalife - an artificial life environment for the evaluation of agent-based systems and evolutionary algorithms for reinforcement learning. In *New Challenges in Applied Intelligence Technologies, Proc. of the IEA-AIE 2008 conference*, Studies in Computational Intelligence Series. Springer, 2008.

[10] P. Maia, E. C. Ferreira, I. Rocha, and M. Rocha. Evaluating evolutionary multiobjective algorithms for the in silico optimization of mutant strains. In *8th IEEE International Conference on BioInformatics and BioEngineering Workshops(BIBE2008).Athens, Greece*, pages 509–514, 2008.

[11] R. Mendes, I. Rocha, E. Ferreira, and M. Rocha. A comparison of algorithms for the optimization of fermentation processes. In *2006 IEEE Congress on Evolutionary Computation*, pages 7371–7378, Vancouver, BC, Canada, jul 2006.

[12] P. Moscato and M. Norman. A 'Memetic' Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 187–194, Amsterdam, 1992. IOS Press.

[13] M. Rocha, P. Cortez, and J. Neves. Evolution of neural networks for classification and regression. *Neurocomputing*, 70(16-18), oct 2007.

[14] M. Rocha, P. Maia, R. Mendes, E. C. Ferreira, K. Patil, J. Nielsen, and I. Rocha. Natural computation meta-heuristics for the in silico optimization of microbial strains. *BMC Bioinformatics*, 9(499), 2008.

[15] M. Rocha, R. Mendes, P. Maia, D. Glez-Pena, and F. Fdez-Riverola. A platform for the selection of genes in dna microarray data using evolutionary algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007), London*, pages 415–421, 2007.

[16] M. Rocha, J. Pinto, I. Rocha, and E. Ferreira. Evaluating evolutionary algorithms and differential evolution for the online optimization of fermentation processes. In *Lecture Notes in Computer Science 4447, Proc. EvoBio2007, Valencia*, pages 236–246, 2007.

[17] M. Rocha, P. Sousa, P. Cortez, and M. Rio. Multiconstrained Optimization of Networks with Multicast and Unicast Traffic. In G. P. el al., editor, *11th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2008)*, volume 5274 of *Lecture Notes in Computer Science*, pages 139–150, Samos Island, Greece, sep 2008. Springer.

[18] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, 1981.

[19] P. Sousa, M. Rocha, M. Rio, and P. Cortez. Efficient OSPF Weight Allocation for Intra-domain QoS Optimization. In *Lecture Notes in Computer Science 4268, 16. Autonomic Principles of IP Operations and Management*, pages 37–48. Springer, 2006.

[20] P. Sousa, M. Rocha, M. Rio, and P. Cortez. Class-Based OSPF Traffic Engineering Inspired on Evolutionary Computation. In *5th International Conference on Wired/Wireless Internet Communications, Coimbra, Portugal, May 2007. Lecture Notes Computer Science*, pages 141–152. Sprinter-Verlag, 2007.

[21] R. Storn and K. Price. Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[22] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman, 2005.

[23] E. Zitzler, M. Laumanns, L. Thiele, et al. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *EUROGEN*, pages 95–100, 2001.